ASSIGNMENT 4: DATA VIZ FOR ADVOCACY

Ivan Aristy — iae225 *CS-GY 6313 B*

1 Interactive Visualization

1.1 Question

How are men's mental states in the US?

The question is a bit general, but I want to communicate to a general audience that is not aware of the high levels of mental health issues that men face in the US.

1.2 Data

1.2.1 Data Source

There are a few data sources to get information from.

The CDC holds lots of information, but particularly I looked into:

1. Behavioral Risk Factor Surveillance System (BRFSS) Survey
2. Household Pulse Survey
3. National Health and Nutrition Examination Survey (NHANES)
4. National Health Interview Survey (NHIS)

However, much to my dismay, lot's of the Data I was looking for was not available. I was able to retrieve information for suicide rates, but the data regarding mental health for men was quite limited.

I ended up usin:g `https://www.cdc.gov/suicide/facts/data.html#cdc_data_surveillance_section_4-suicide-rates` for suicide rate information, and `https://www.apa.org/monitor/2015/12/numbers` for general information and stats regarding suicide rates.

1.2.2 Serving Information

We are using FastAPI to expose the data to the frontend. To accurately model the data, I created the following interface:

```
class ChampionInstance(BaseModel):
  name: str
  patch: float
  win_rate: float
  win_rate_delta: float
  modified_winrate: float
  pick_rate: float
  tier: str
  rank: int
  ban_rate: float
  games: int
```

and exposed the following function to the frontend:

```
@app.get("/test/{champion_name}&{patch_version}")
async def test(champion_name: str, patch_version: str):
  testChampion: ChampionInstance =
    await api.get_champion_data(champion_name, patch_version)
  return testChampion
```

All the functions used were made asynchronous, since load times were a complaint I received from my peers while showing them the assignment.

The benefit of using live data is that, whenever a new patch is released, the user can see how their champion has been impacted by the patch. Additionally, for the current patch, changes in data are reflected in real time.

1.3 Visualization

1.3.1 Frontend Setup

The frontend is a simple React application that uses the `D3` library to create the chart, and react hooks to update and keep track of state (dynamic reloading of data depending on user selected parameters).

For the main App component, we define a useEffect hook that will achieve multiple "Quality of Life" improvements for the user:

- Updating the chart whenever the user selects a new champion, metric, or patch range.
- Displaying a loading spinner while the data is being fetched.
- Asynchronous fetching of data to speed up the loading process.
- Invalidate erroneous data but keep plotting the chart.

```
useEffect(() => {
    async function fetchData() {
      setLoading(true);

      const patchRange = patches.slice(
        patches.indexOf(startPatch),
        patches.indexOf(endPatch) + 1
      );

      try {
        const results = await Promise.all(
          patchRange.map(async (patch) => {
            const response = await fetch(
              `http://127.0.0.1:8000/test/${selectedChampion}&${patch}`
            );
            if (!response.ok)
              throw new Error(`Failed to fetch data for patch ${patch}`);

            const result = await response.json();
            return { patch, value: result[selectedMetric] };
          })
        );

        setData(results);
      } catch (error) {
        console.error("An error occurred while fetching data:", error);
      } finally {
        setLoading(false);
      }
    }

    setData(null);
    fetchData();
  }, [selectedChampion, selectedMetric, startPatch, endPatch]);
```

1.3.2 Interactivity Logic

To allow for interactivity, we create a few drop down labels that allow the user to select the data that they want to plot. For example, here is the code for the "Champion" Label:

```
<label>
  Champion:
  <select
    value={selectedChampion}
    onChange={(e) => setSelectedChampion(e.target.value)}
  >
    {champions.map((champ) => (
      <option key={champ} value={champ}>
        {champ}
      </option>
    ))}
  </select>
</label>
```

This also allows you to type out the champion name instead of going through every single champion in the dropdown.

Additionally, I implemented another QoL feature that deals with situations where the user selects a patch range that is not valid. For example, if the user sets start patch to 14.20 and end patch is at 14.10, the code will automatically adjust the end patch to 14.20.

1.3.3 Visualization Logic

As mentioned previously, I used D3 to create the chart. For my previous assignments I have used D3's sister library, Plot, which is a wrapper around D3 that makes it easier to create charts. Using D3 directly was a lot more challenging, and I concluded on using Plot for future projects with D3 on a case-by-case basis.

Nevertheless, D3 made me consider the components of a chart in a more explicit manner:

- Margins: Focused on creating ample space for the chart.
- Scales: Dynamically mapped the changing domain of the data to the range of the chart, and accounted for "edge" metrics. (For example, rank is plotted inversley, since a lower rank is better... Rank 1 > Rank 2).
- Line: Simple mapping of values received by the backend to the chart.
- Tooltip & Circles: Created dynamic circles that would show the value of the data when hovered over, as well as grow slightly. Also considered circle size and color for visibility.
- Axes: Created axes, gridlines, and basic labels for intuitive and fast understanding by the user.

1.4 Improvements

1.4.1 Champion Image

A major piece of feedback was to include the champion image in the visualization. This would allow the user to quickly identify the champion they are looking at, not having to rely on the champion name alone, which could be pretty small on some devices.

This was actually added.

1.4.2 Small Patch Ranges

We have a degenerate case where the user selects a patch range that is so small that visualizing it in a line chart is not very useful.

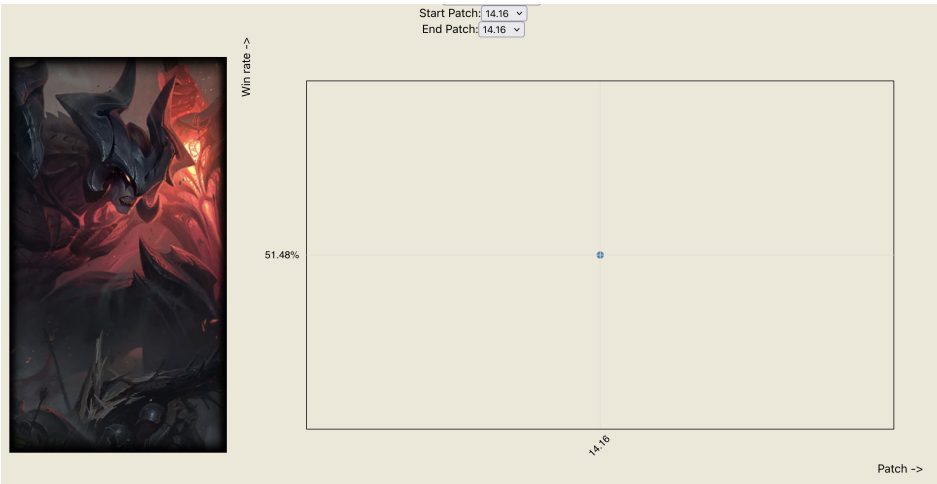

Figure 1: Solo patch range.

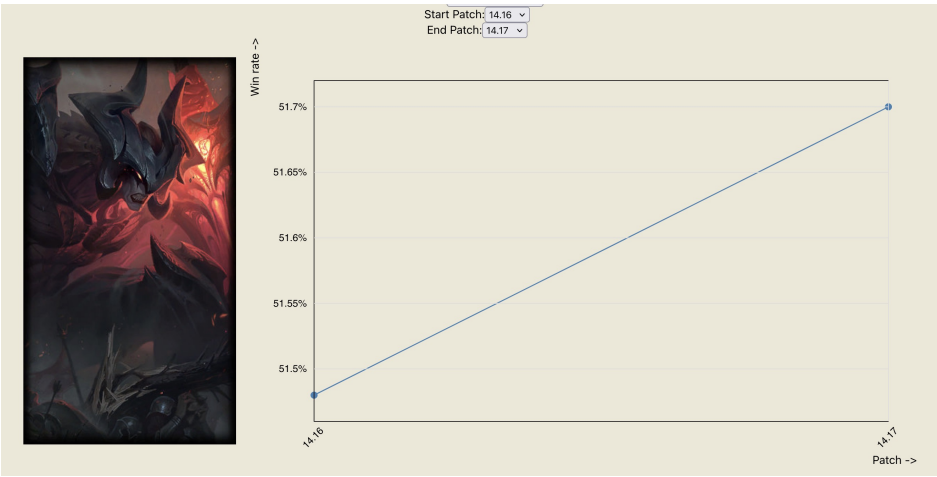

Figure 2: Size 2 patch range.

See Figure 1 and Figure 6 for an example of this.

I did not create a fix for this specific situation, since the Visualization is for users to see how their champion has been impacted over time.

1.4.3 Lane Specific Data

Another piece of feedback was to include lane specific data.

Why? Well, a botlane champion will have a different winrate if they are played toplane. In layman's terms, the best pizza is not a very good burger. Also, some champions are played in multiple lanes, so it would be interesting to see how their winrate changes depending on the lane.

I deemed this to be beyond the scope of the assignment, but it is a good idea.

1.5 Conclusion: Do we answer the question?

I believe for both:

1. How has my champion's win rate changed over time?

2. Is my champion still strong in the current meta?

We more than answer the question. By providing additional data like rank, winrate delta, and games, we allow the user to make a more informed decision on whether their champion is strong or not.

Corki was just buffed on Wednesday (and I hate that), do these images answer both questions?

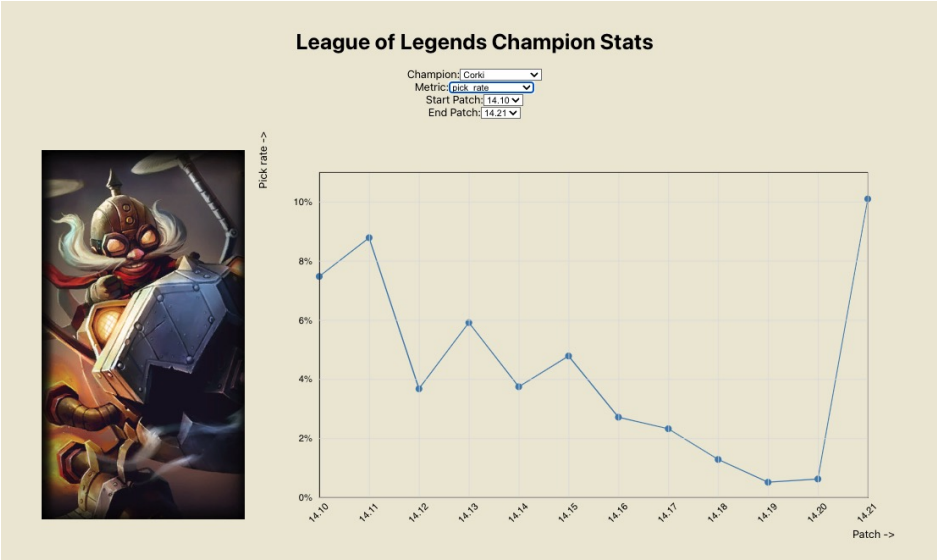

Figure 3: Corki Pick Rate

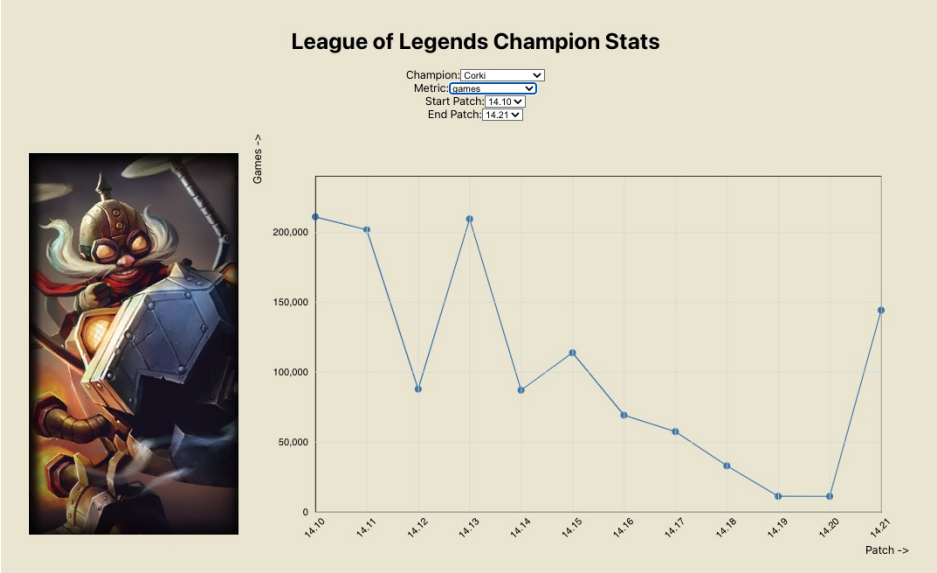

Figure 4: Corki Games Played

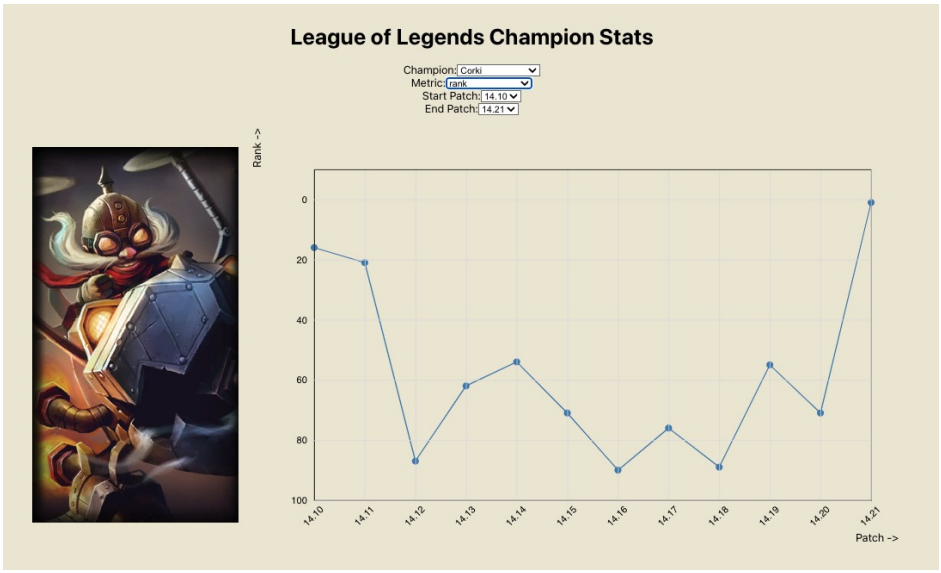

Figure 5: Corki Rank

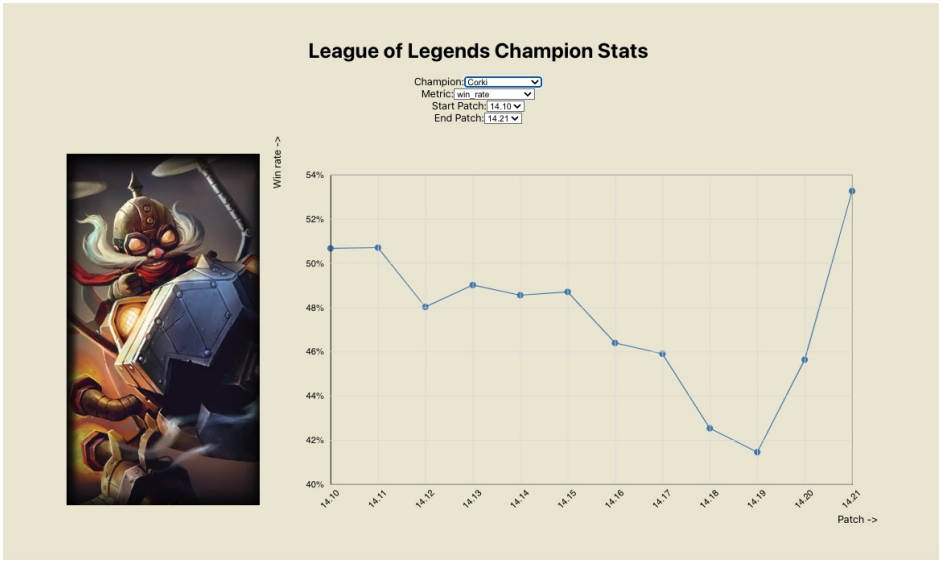

Figure 6: Corki Winrate